Hybrid FPGA/Multi-core CPUs for Industrial Applications

Robbie Vincke, Arno Messiaen and Jeroen Boydens

Abstract – **The rising demand for high-performing embedded systems made FPGAs ubiquitous. Combining the strengths of an FPGA and a general purpose processor on one chip does not only simplify PCB layout, it also removes the communication bottleneck between processor and FPGA. Moreover, it allows the designer to partition applications and map parts of an application to either programmable logic or processing system. A case study on a voice-over-ethernet system illustrates partitioning by means of the processing requirements. This partitioning is called the Planar Design Pattern. Management of the communication channel is done by one of the processor cores. While high-speed data streaming itself is done in programmable logic.**

Keywords – **Hybrid FPGA/Multi-core, Planar Architectures, voice-over-ethernet**

I. Introduction

High-performing embedded systems are mostly equipped with massive parallel reconfigurable devices such as FPGAs. On the one hand, by means of reconfiguration possibilities FPGAs provide flexibility while offering ASIC-like performance. Especially for signal processing the usage of FPGAs is a huge benefit. On the other hand, general purpose CPUs provide good control functionality and are easier to program. By combining both, an application can be partitioned in such way it benefits as well the strengths of FPGAs as CPUs.

Previously, embedded designers often used general purpose processors in combination with off-chip FPGAs. In this configuration, the FPGA can offload the CPU by calculating computational intensive algorithms. However this solution needs additional PCB layout effort. Moreover the communication channel between FPGA and CPU has limited bandwidth.

Recently, the combination of FPGAs and general purpose processors on a single chip is finding its way towards industry. The interconnection between programmable logic and processor is reaching 4 to 8 GB per second, which means communication between both technologies is not an issue anymore. Other advantages of migrating to this hybrid architectures are reduction of the bill of materials (BOM), since only one chip is needed. This also affects the PCB board space. No off-chip interconnection between programmable logic and processor is needed, which in turn reduces the possibility of EMC problems.

Robbie.Vincke@khbo.be is a scientific staff member at KHBO funded by IWT-110174.
Arno Messiaen is an electronics engineering graduate at KHBO.
Jeroen.Boydens@cs.kuleuven.be is a professor at KHBO and an affiliated researcher at KU Leuven, dept. CS, research group DISTRINET.

The combination of programmable logic and general purpose processors on a single chip also provides some great new possibilities. First the incorporation of a high performance general purpose processors provides the ability of running a standard operating system, which is optimal for user interaction and programming flexibility. This means legacy software can be executed on a one of the cores of the hybrid architecture. On the other hand including programmable logic provides flexibility for custom interfaces and a potential performance gain as a result of intrinsic massive parallel behavior.

In Section II three possible use cases are given for hybrid FPGA/Multi-core CPUs. In Section III a short overview of the currently available platforms is given. Section IV describes the usage of the Planar Pattern, while Section V applies the pattern on an industrial case study. Section VI concludes the paper.

II. Use Cases

One of the possibilities of combining programmable logic and a general purpose processor is using programmable logic as hardware accelerators (Figure 1.A). This means some computational intensive operations can be outsourced to programmable logic to increase system performance, thus offloading processor. Typical applications are offloading of DSP-functions to programmable logic.

Not only can specific functions be outsourced to programmable logic. Programmable logic can also serve as an independent subsystem (Figure 1.B). Real-time applications for example can be completely outsourced to programmable logic, while the processor system continues executing non-real-time applications. By applying this strategy, a planar architecture is created containing a slow path and a fast path. This can be useful in networking applications where the fast path contains packet forwarding operations and the slow path contains control functions.

Besides outsourcing computational tasks to programmable logic, programmable logic can also be seen as a way of adding custom interfaces to the system or adding additional peripherals to the system (Figure 1.C) besides the standard available interfaces on de the processor subsystem.

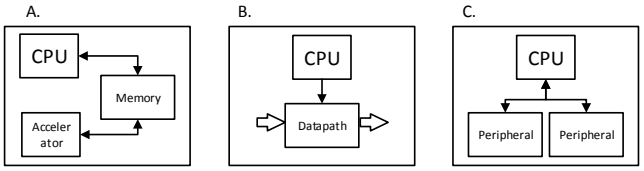

Figure 1. Use Cases of Hybrid FPGA/Multi-core Architectures

III. COMMERCIALLY AVAILABLE PLATFORMS

Today, *Xilinx* and *Altera* each provide hybrid FPGA/CPU solutions. Both combine 28 nm technology FPGA fabric with a dual core ARM Cortex A9 general purpose processor. Interconnections between FPGA fabric and processing system is implemented in both cases using the AMBA AXI protocol [1]. However there are some major differences between both systems.

On the one hand, the *Xilinx* solution (Zynq-7000) has a processor-centric approach. This means the processing system is always the first to start up. In second line the FPGA subsystem can be started. This means FPGA functionality cannot exist without a running processing system [2].

On the other hand the *Altera* solution (AlteraSoC) can boot and operate in three ways [3]: (1) identical to the *Xilinx* solution, The FPGA subsystem can be started through the processing system (Figure 2.A). (2) The other way around the processing system can be started through the FPGA subsystem (Figure 2.B). (3) Both subsystems can boot and operate entirely independent (Figure 2.C). This last configuration is a major advantage for safety-critical systems.

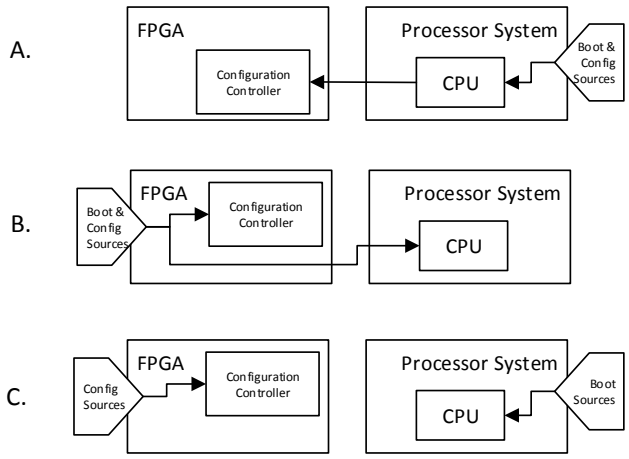

FIGURE 2. BOOT SEQUENCES OF ALTERASOC[4]

A second big difference between both competitors is the debug capability. *Altera* provides support for cross-triggering. This means a software breakpoint in the processing system can halt the FPGA fabric and vice versa. Good debugging support can improve time-to-market drastically. *Xilinx* architecture provides separate debugging for the processor system (GDB debugging) and for the programmable logic (internal logic analyzer such as Chipscope). Both debugging planes do not interact with each other.

IV. PLANAR ARCHITECTURE

As mentioned in Section II, several use cases are possible for these hybrid architectures. Since two subsystems with different strengths and performance capabilities can be combined, application performance optimization by means of partitioning is possible. It is necessary to determine which part of the application can benefit from the processor subsystem or the FPGA subsystem. Since the currently available platforms consists of a dual core processor combined with FPGA, it is possible to separate three *"planes"*. This is called a Triple Planar Architecture (TPA). To accommodate three planes the processor needs to be configured as an Asymmetric Multiprocessing system (AMP) instead of a Symmetric Multiprocessing system (SMP).

In AMP systems each CPU core runs another instance of an operating system. In practice the used operating system is matched to the performance requirements. Planes with real-time requirements are accommodated with a Real-time operating system (RTOS). AMP systems are commonly used in telecommunication systems where a subset of the cores control the communication channel and the remaining cores (with higher performance requirements) are exclusively used for packet forwarding, etc.

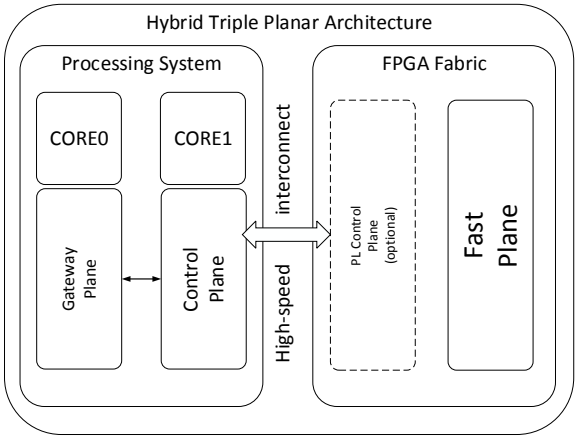

FIGURE 3. SCHEMATIC OVERVIEW OF A HYBRID TRIPLE PLANAR ARCHITECTURE

As illustrated in Figure 3, the planar architecture uses one core as a user interaction core (gateway plane) just to provide the needed interface to receive requests from the user. The other core is used to process such requests and delegate commands to the executing plane (fast plane). For example sensor data, measured by the fast plane, can be send to the user by forwarding it from the fast plane to the control plane, which converts the data into the desired format, and in turn forwards it to the gateway plane. The gateway plane encapsulates the data into the desired format (Ethernet, etc.). If the programmable logic should implement a more control-like function the fabric can implement additional soft-core processors such as *MicroBlaze* or *Nios II*. The FPGA fabric can implement a soft-core processor too if it benefits the application.

The main force of TPA for use in industrial applications is the strict separation of the planes. This can be beneficial in safety-critical embedded systems. Errors on either one of the planes should not affect the proper functioning on another plane.

V. CASE STUDY: VOICE-OVER-ETHERNET

To start our case study, a voice-over-ethernet implementation on a standalone processor was studied. It introduced performance hick-ups due to a massive amount of interrupts on the MAC peripheral. Nevertheless, it is relatively simple to let a processor react on management requests over ethernet from external modules. One of the

solutions is leading all ethernet traffic through an FPGA. The MAC peripheral is now connected to the FPGA instead of directly to the ethernet PHY (physical transceiver). The FPGA only forwards management packets (commands from external modules) to the MAC peripheral of the processor. The audio data itself will be transferred using the RTP protocol (Real-time Transfer Protocol) [5]. Subsection (A) describes the used development platform. In Subsection (B) the setup of processing system is explained, in Subsection (C) the usage of the programmable logic is explained.

A. Platform

The hybrid implementation of the voice-over-ethernet system will be done on a Xilinx ZedBoard [6] featuring a Xilinx Zynq-7000 SoC XC7020 hybrid FPGA/ARM Cortex-A9 multi-core processor. 512 MB of DDR3 RAM is available on the board. The onboard Ethernet PHY is hardwired to the processing system of the Zynq. Therefore an add-on board is needed to connect an ethernet PHY to programmable logic. The Avnet ISM networking FMC add-on board [7] is used.

B. Processing System

The processing system will be configured in an AMP fashion. The master CPU core0 will run a general purpose Linux operating system and the slave CPU core1 will run bare metal applications (without operating system). If the slave processor would need shared peripherals it should send a request to the master processor. However, by partitioning the application properly, the slave process (control plane) never needs any peripherals. To prevent any unauthorized accesses to memory of another CPU core the master core is configured exclusive access to certain memory addresses. The same is true for the slave core. Since both CPU cores share L2 cache, the usage of L2 cache is disabled in the slave core. Communication between both planes is done using a small segment of shared memory.

C. Programmable Logic

The programmable logic is used to implement the partition with the highest performance requirements: the audio gateway. The audio gateway implements the interface between the Ethernet PHY (DP83640) and the audio codec (ADAU1761) and is bidirectional (Figure 4).

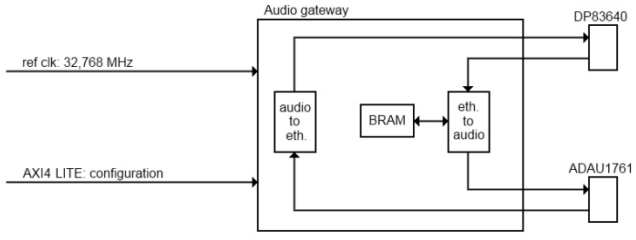

FIGURE 4. VOICE OVER ETHERNET AUDIO GATEWAY IN PROGRAMMABLE LOGIC[8]

Communication with the Ethernet PHY is done using the Media Independent Interface (MII). The Ethernet PHY itself is configured through the Management Data Input/Output (MDIO). The interface to the audio gateway is an AMBA AXI4 Lite interface. The audio-to-ethernet part consists of an I²S interface, clock domain crossing between codec clock and MII clock, sample FIFO, RTP generator state machine and packet FIFO as showed in Figure 5.

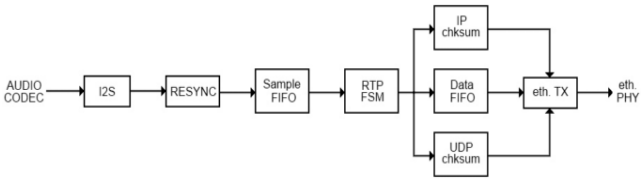

FIGURE 5. AUDIO-TO-ETHERNET INTERFACE[8]

The ethernet-to-audio part adds BRAM memory as buffer for unmatched input and output clocks as illustrated in Figure 6.

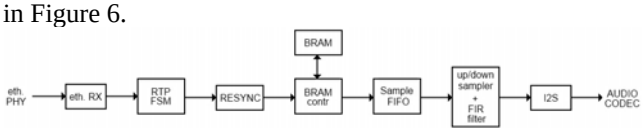

FIGURE 6. ETHERNET-TO-AUDIO INTERFACE[8]

VI. CONCLUSION

The case study on voice-over-ethernet illustrates the partitioning process for migration to a hybrid FPGA/multi-core system using the latest Xilinx Zynq-7000 platform. Partitioning is done by means of processing requirements where low processing power partitions are mapped to the processing system and high-performance data streaming is mapped to programmable logic. Besides the learning curve for asymmetric configuration of the multi-core processor and interfacing to the programmable logic using the AXI-bus the planar hybrid architectures provides at better performance in comparison to an off-chip FPGA. Additionally PCB board layout is simpler and BOM reduces.

ACKNOWLEDGMENT

The authors wish to acknowledge Jürgen Lambrecht and Televic Rail NV for providing the experiments with the voice-over-ethernet system on a hybrid core.

REFERENCES

[1] “AMBA AXI Protocol Specification.” ARM, 2003.
[2] Xilinx, “Zynq-7000 All Programmable SoC - Technical Reference Manual,” vol. 585, 2013.
[3] Altera, “AlteraSoC: Booting and Configuration,” vol. 3, no. November. pp. 1–24, 2012.
[4] Arrow, “Altera’s SoC FPGAsYour User-Customizable System on Chip.” 2013.
[5] *Network Protocols Handbook*. Javvin Technologies Inc., 2005, p. 340.
[6] “Zedboard overview.” [Online]. Available: http://www.zedboard.org/content/overview.
[7] Avnet, “ISM Networking FMC Module.” [Online]. Available: http://www.em.avnet.com/en-us/design/drc/Pages/ISM-Networking-FMC-Module.aspx.
[8] A. Messiaen, “Kritische evaluatie van een Xilinx Zynq System-on-Chip implementatie aan de hand van een gevalstudie voor een Human Machine Interface,” KHBO, 2013.